ABSTRACT OF THE DISCLOSURE

Security mechanisms detect and intervene in a malicious attack against a runtime function, even in the presence of a coding flaw such as a buffer overrun or overflow. One such exemplary mechanism uses a predetermined security list of the valid targets for a first runtime function (such as longjmp). For every call to a second runtime function (e.g., setjmp) that prepares for a later invocation of the first runtime function, the dispatcher finds and stores a reference to this list. When a subsequent attack targets the runtime functions by creating an attacker-provided setjmp target address (e.g., the attack overwrites the longjmp target address so that the pointer points somewhere else, such as code provided by the attacker or code that already exists that will eventually pass control to code provided by the attacker), the new (attacker provided) target address is compared to a reference list of the real (valid) target addresses. The list of real target addresses is stored in memory. If the target address that has been provided is found on the reference list, then the runtime function (e.g., longjmp) is allowed to continue to execute by the dispatcher (which may be the actual runtime function). Otherwise, the dispatcher assumes the application is under attack and terminates the process' execution.